CODE WIZARDS

Hathora

Whitepaper

13 October, 2025

Presented to:
Harsh Pandey
Gabi Weinberg

Presented by:
Carl Ind

Table of Contents

Confidentiality Statement

This document is confidential and intended solely for the use of the individual or entity to whom it is addressed. If you are not the intended recipient, please notify us immediately and delete this document. This document contains proprietary and confidential information and may not be disclosed, reproduced, or distributed to any third party without the prior written consent of Code Wizards.

All prices and fees mentioned in this document are exclusive of VAT, unless otherwise stated. The client shall be responsible for paying all taxes, duties, and levies that may be applicable on the services provided by Code Wizards, including VAT.

The information contained in this document is provided "as is" without any warranty, express or implied, as to the accuracy or completeness of the information or the suitability of the services proposed herein. Code Wizards shall not be liable for any damages whatsoever arising out of or in connection with the use of this proposal document or the services proposed herein.

This document and any accompanying documentation are the property of Code Wizards and may not be used, copied, or distributed for any purpose other than to evaluate the services proposed herein. By accepting this document, the client agrees to be bound by the terms and conditions set forth herein.

Code Wizards reserves the right to modify or withdraw this document at any time without prior notice. Any modifications or changes to this proposal document shall be communicated to the client in writing.

1 Executive summary

When developers evaluate multiplayer infrastructure there are a large number of options. Hathora presents an approach that differs from other providers often designed for broad infrastructure management. The platform is built specifically for game server hosting, placing an emphasis on developer experience and operational simplicity rather than extensive infrastructure customisation. It runs on external cloud providers and bare metal, with orchestration abstracted behind a rooms model that aligns with common multiplayer patterns.

The evaluation demonstrated a direct process for moving from initial setup to a functional multiplayer server deployment. The platform is designed to be accessible, including for developers who do not specialise in infrastructure tasks. At the same time, teams should note current limitations: they have launched titles with ~100,000 CCU across platforms, but no ultra CCU launches at the time of writing, limited deep system tuning and custom networking or storage controls.

For developers considering alternatives to highly configurable game server orchestration systems or self managed cloud infrastructure, Hathora provides a managed solution focused on a common developer workflow. The hybrid model, which combines a bare metal baseline with cloud overflow, can reduce costs when baseline utilisation stays high and bursts are short, with pricing based on active vCPUs and outbound bandwidth. Security protections apply across both capacity types and are managed automatically within Hathora.

While its feature set may not be optimal for every project, it is intended to streamline the operational processes for a wide array of multiplayer games, making it a strong choice for teams looking to reduce time to market with its solid baseline of features. While not proven at massive scale Hathora's features offer a kickstart to any team wanting to host dedicated game servers and is a solid contender in the orchestration and scaling market.

2 Introduction

Multiplayer game development faces a persistent infrastructure challenge. Developers prefer focusing on gameplay, networking code, and player experience rather than container orchestration, load balancers, and scalers.

Game server provision platforms often require developers to become part-time DevOps engineers just to deploy basic multiplayer functionality. They favour tuning and customisation over simplicity and ease of use.

For those games studios without DevOps skills this can mean time lost spent configuring services, debugging networking issues caused by firewall misconfigurations, or facing unexpected server costs due to misconfigured scaling. These infrastructure problems divert attention from actual game development priorities.

Hathora gained attention by promising to address this specific pain point. Rather than offering a cloud or bare metal specific solution with high customisability, they built a solution specifically for game developers which is easy to use. This evaluation aims to determine whether this game-focused approach delivers meaningful benefits or simply represents clever marketing around familiar hosting complexities.

3 Hathora's Approach

Hathora centres on "rooms," essentially game server instances that scale in line with demand when players need multiplayer sessions. This design choice aligns well with common multiplayer game patterns: players join lobbies, get matched or create rooms, and connect to dedicated server instances for their sessions.

The underlying technology uses Docker containers and orchestration, but this complexity remains highly abstracted. Developers provide a Dockerfile, configure basic settings, and Hathora handles the orchestration. Containers run across multiple regions, scale automatically based on demand, and terminate when rooms become empty.

The key distinction lies in the game-specific workflow focus. Rather than configuring generic compute instances, developers configure game rooms. Instead of managing load balancers, they manage regional distribution to optimise player latency. The abstractions are designed to simplify the challenges of multiplayer game development.

4 Our approach to testing

We conducted comprehensive testing across various deployment scenarios to evaluate Hathora's performance in real development workflows. This included deploying an Unreal Engine Lyra server, a JavaScript-based multiplayer game with custom lobby service integration, and experimenting with Unity SDK client integration.

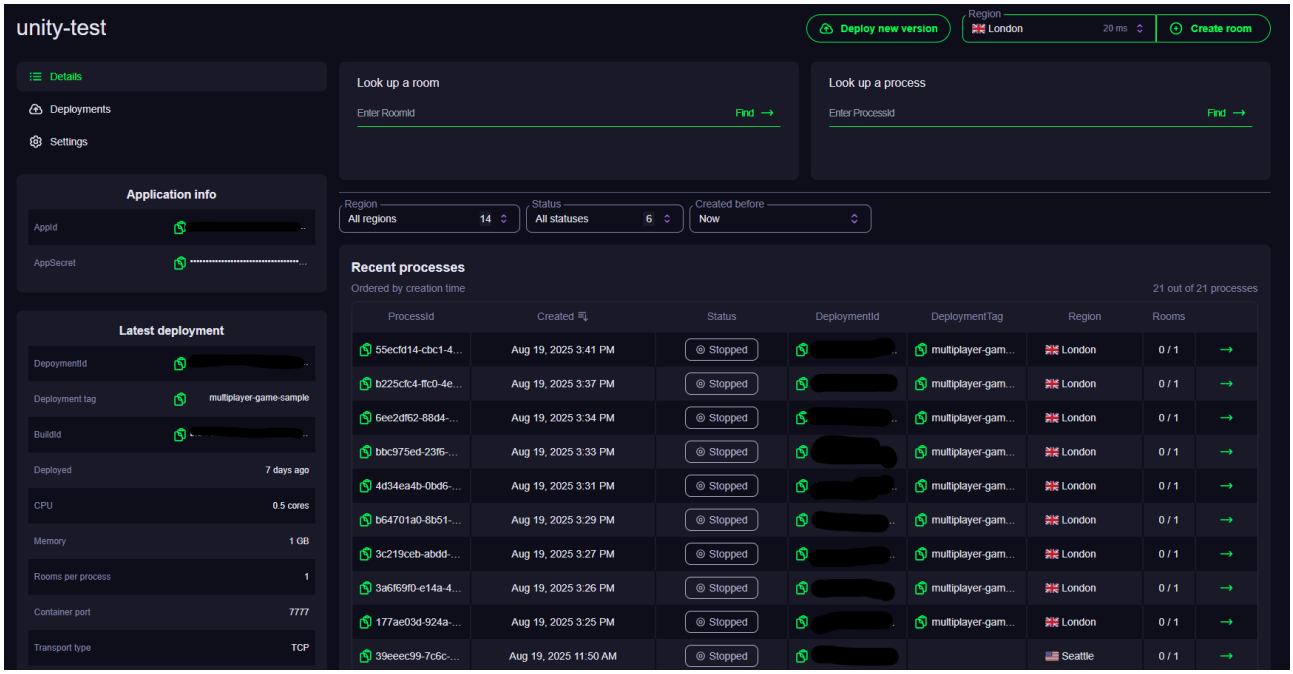
Our objective was to validate developer experience claims through practical testing. We examined whether deployment is simplified, as advertised, and how the platform manages common development challenges, such as frequent redeployments, debugging connection issues, and monitoring server performance under load. Testing spanned multiple regions, simulated varying player loads, and involved deliberate attempts to identify failure modes.

We maintained a sceptical perspective, focusing on thorough documentation review, edge case testing, and cost comparisons with equivalent infrastructure alternatives. With any claimed time and complexity savings requiring substantial proof to be included.

However some elements are deliberately out of scope as they're hard to repeat and validate independently. These are largely in the areas of proactive and reactive 24x7x365 support from Hathora teams and validating behaviour of Hathora with poor game server implementations or silent crashes.

5 Benefits/Use cases

The deployment experience exceeded expectations. Initial server deployment took about 10 minutes with a container ready to go, or roughly 1 to 2 hours to build and containerise a sample from scratch. Hathora tell us that they also offer engineering support for the onboarding and initial setup. On competitor platforms, setting up an equivalent multiplayer server typically takes 2 to 4 hours just for networking configuration.



For teams new to containerisation or facing deployment challenges, Hathora offers direct onboarding support through scheduled calls with their technical team. These sessions, typically lasting around 45 minutes, provide hands-on assistance with containerising game server builds and walking through the deployment process. This personalised support significantly reduces the learning curve for teams unfamiliar with Docker workflows or multiplayer infrastructure deployment.

For production deployments requiring guaranteed response times, Hathora provides a 24/7 support package with a 30-minute SLA for engineer response. This level of support addresses the critical need for rapid incident resolution in live multiplayer games. Additionally, the platform maintains a public server status tracker (https://hathora.instatus.com/) that provides real-time visibility into service health across all regions, enabling teams to quickly distinguish between game-specific issues and platform-wide incidents.

The "room" abstraction functions effectively. Developers focus on game sessions rather than server instances and load balancing. Testing scenarios are intuitive: spin up a room to

test new game features, then use room IDs to track down specific sessions when players report issues. This aligns naturally with existing multiplayer architecture.

Regional deployment simplifies typical complexities. Competitors require setting up and managing their network configurations and complex networking for multi-region deployment. Hathora automates this: developers select desired regions, and rooms provision in optimal locations for each player group.

Network performance across regions consistently meets gaming requirements. Testing revealed latency under 50ms for in-region connections (i.e., nearby players connecting to their local region) with minimal packet loss, suitable for most multiplayer genres including competitive titles. The platform's regional distribution effectively optimises player routing, though teams should validate performance for latency-critical applications requiring sub-20ms response times.

Scaling performs as expected from modern infrastructure. Player demand spikes trigger new servers within thirty seconds during testing. Demand reduction leads to automatic resource cleanup without incurring idle capacity costs. Major capacity increases still require the usual 2-3 minute delay, but normal fluctuations feel responsive.

The logging and monitoring approach alleviates operational burdens. Game server logs capture automatically and remain accessible through the web console and API. No complex log management configuration or aggregation setup is needed. You can access logs for any running or recent server. For teams without dedicated DevOps resources, this significantly reduces operational overhead.

For deeper troubleshooting needs, SSH access is available when required, providing developers with direct server access for debugging complex issues. The setup process involves SSH key configuration, port management, and Dockerfile modifications, which maintains security while ensuring access is intentional and properly configured. This capability bridges the gap between simplified operations and detailed debugging when necessary.

Incident response integration operates through programmatic monitoring APIs that enable external alerting systems. Teams can build custom monitoring workflows using the Fleet and Process metrics endpoints to trigger PagerDuty or similar incident management platforms when game-specific thresholds are breached. For example, monitoring room creation failure rates, regional capacity exhaustion, or abnormal process termination patterns through automated polling of GetFleetMetrics and GetProcessMetrics endpoints. This approach allows teams to establish game-aware alerting rather than relying solely on generic infrastructure metrics, ensuring that incidents affecting player experience receive appropriate escalation and response.

As for support Hathora offer 24/7 365 Support with a 30 minute SLA for a engineering response, however we did not test this feature during our review of the overall service.

6 Risks/Considerations

The primary concern is operational lock-in, which differs from traditional vendor lock-in. While game server code remains portable through containerisation/Image registries, operational knowledge and tooling become specific to Hathora. Room management, deployment pipelines, and monitoring workflows require rebuilding if you migrate elsewhere.

Customisation options are limited compared to self-managed infrastructure. Optimising the underlying OS or networking stack remains impossible. However, custom monitoring agents can be deployed within game server containers for log export to external systems like S3, Datadog, and Honeycomb. For most games, these limitations are acceptable, but specific performance requirements or deep system access needs may face restrictions.

Regional coverage offers good but incomplete global reach. Fourteen regions fall short compared to major competitors' thirty-plus regions, which could be problematic for truly global games. Additionally, tier restrictions limit access to advanced capabilities since dedicated hardware and enhanced security features require Enterprise pricing.

Capacity and scaling characteristics are designed for cost efficiency and responsiveness. Room termination occurs immediately when emptied, avoiding unnecessary costs, while new room provisioning completes within 30 seconds under normal conditions. Major capacity scaling during traffic spikes requires 2-3 minutes, though this can be mitigated through pre-scaling configurations and cloud baseline settings. Regional capacity limits are designed to handle typical multiplayer game loads, with overflow and multi-region distribution capabilities for larger deployments. These scaling patterns are optimised for most multiplayer game traffic profiles, with configuration options available for titles with specific scaling requirements.

Hathora specifically calls out that there are additional scaling options that deal with the different scaling strategies mentioned above, such as min cloud node scaling, which essentially means that an event is expected to scale through bare metal capacity and you are extending the buffer by scaling cloud capacities. As well as scaling over arching thresholds that can be tweaked to combat and smooth out spikes in game server traffic. The fact that these tools exist highlights the complexities of operating game servers in today's video game market.

Cost predictability improves over competitors for typical game workloads, but the hybrid pricing model adds complexity. Baseline capacity commitments combine with uncertain bursting costs during traffic spikes, necessitating careful capacity planning.

A major component of any game server solution is how it handles problems outside of normal operational hours and this could not be validated as part of this evaluation. If

worldwide utilisation of your game requires global support then this should be agreed and tested before choosing Hathora as a platform.

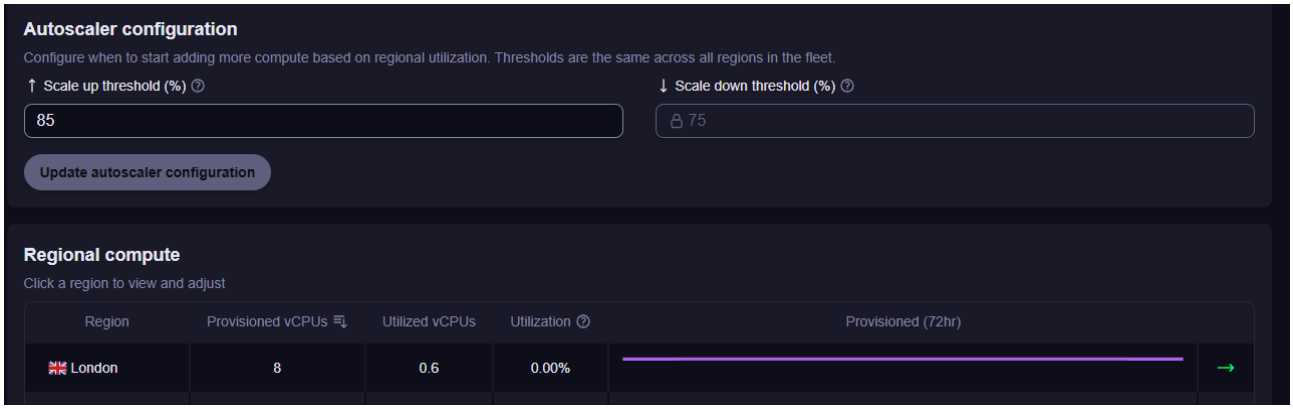
7 Fleets: Enterprise differentiation

Fleet management sets Hathora's enterprise offerings apart from basic container hosting. Pro and Enterprise customers benefit from dedicated infrastructure across multiple cloud providers and bare metal hosts instead of shared hardware. Hathora runs on external cloud infrastructure and does not operate its own in-house cloud. The platform spans multiple external providers and bare metal and can choose per region based on price, capacity, and latency goals; in typical setups a single provider backs a region at a time, with the ability to switch or burst when economics or capacity change.

For enterprises with existing cloud commitments, Hathora offers a Bring Your Own Cloud service that allows deployment into customer-owned cloud tenants, enabling teams to leverage existing volume discounts and minimum spend commitments while maintaining Hathora's orchestration and management capabilities.

What impressed most about the fleet experience was how completely automatic it felt from a developer perspective. Despite the underlying complexity of managing dedicated nodes across multiple regions, the deployment process remained identical to the basic tier. No additional configuration, fleet-specific commands, or infrastructure management was required because rooms provisioned on our dedicated London hardware transparently. This seamless transition from shared to dedicated infrastructure eliminates the operational overhead typically associated with enterprise-grade hosting.

The hybrid infrastructure approach showcases innovation. Committed capacity runs on cost-effective bare metal servers, while demand spikes automatically provision additional capacity from cloud providers. Testing revealed seamless transitions from a developer perspective, although the two-minute cloud scaling delay necessitates careful baseline capacity planning.



The operational intelligence level impressed during evaluation. The scaler responds to game server allocation patterns rather than simple CPU metrics, ensuring optimal utilisation across the fleet. Server distribution occurs efficiently across regions, and underutilised capacity is reclaimed aggressively without affecting active games.

Monitoring and fleet management tools are tailored for game infrastructure. Instead of generic metrics, the interface displays game-relevant data: room allocation rates, regional utilisation patterns, and capacity planning projections. The Fleet API supports sophisticated automation scenarios, including pre-scaling for scheduled events and external matchmaking system integration.

Fleet management capabilities are available across all paid plans, with only the free tier scheduling on multitenant shared infrastructure. This accessibility ensures that teams can benefit from dedicated fleet resources without requiring enterprise-level commitments, though advanced features and larger capacity allocations scale with pricing tiers.

Advanced scaling controls provide fine-tuned capacity management for demanding scenarios. Teams can configure baseline settings to pre-scale minimum nodes, particularly valuable for known launch events where large traffic spikes are expected. The scale-up threshold configuration (defaulting to 85% utilisation) allows adjustment of autoscaler eagerness, letting customers choose their availability versus cost trade-off. These controls enable sophisticated capacity planning that balances responsiveness with economic efficiency.

8 Observability and monitoring capabilities

Hathora's observability infrastructure addresses a critical concern for production multiplayer deployments: maintaining visibility into distributed game server behaviour at scale. The platform provides comprehensive monitoring without requiring teams to implement custom telemetry solutions.

The process identification system is particularly valuable for production troubleshooting. External match identifiers can be linked to room creation, allowing rapid correlation between player reports and specific server instances. This functionality integrates through both the Hathora Console and programmatic API access via GetRoomInfo endpoints, simplifying the complex task of connecting external matchmaking systems to running infrastructure.

Real-time metrics collection operates automatically across fleet utilisation, scaler behaviour, and individual process resource consumption. For AAA operations, this granular visibility into CPU, memory, and network utilisation patterns enables capacity planning and performance optimisation. Metrics access is available through both dashboard interfaces and programmatic endpoints (GetProcessMetrics, GetFleetMetrics), supporting integration with existing monitoring workflows.

Enterprise-grade Prometheus integration is a significant capability for larger operations. Prometheus Remote-Write compliance allows consolidation of Hathora infrastructure

metrics with external systems like matchmakers, backend services, and player analytics platforms. This integration enables unified observability dashboards and sophisticated alerting workflows essential for AAA production environments.

Log management addresses another critical operational requirement. The platform captures all stdout and stderr from running processes, providing complete visibility into game server behaviour. Log access operates through multiple interfaces: real-time console streaming, programmatic API access, and CLI integration via hathora log commands. Importantly, logs remain accessible for both active and terminated processes, crucial for post-incident analysis.

Flexible log export capabilities support enterprise integration requirements and maintain deployment portability through standard Docker registries and containerised workflows. Teams can deploy custom agents within game server containers to export logs to external systems: cloud storage (S3), observability platforms (Datadog, Honeycomb), or proprietary logging infrastructure. Monitoring solutions implemented directly within containers remain portable across different hosting environments, while full metrics export to customer Prometheus instances eliminates dependency on Hathora APIs or Console for process metrics. This approach ensures monitoring data remains accessible through existing infrastructure while meeting compliance and operational requirements.

Data retention policies balance operational needs with cost considerations. Standard 72-hour retention provides adequate time for routine troubleshooting and performance analysis. Enterprise customers can negotiate extended retention periods for compliance, long-term trend analysis, or detailed post-mortem investigations.

The developer experience prioritises usability without sacrificing functionality. Console interfaces offer intuitive real-time monitoring, while comprehensive API access enables automated monitoring and alerting integration. This dual approach supports both hands-on debugging and enterprise-scale operational automation.

9 SDK integration and API ecosystem

Hathora's integration architecture uses progressive enhancement, allowing teams to begin with minimal Docker-based deployment while accessing advanced capabilities as needed. This approach is especially valuable for enterprise development, where initial prototypes may require simple solutions, but production deployments need sophisticated features.

The core SDK capabilities address essential multiplayer infrastructure needs without requiring major architectural changes. Room management APIs offer programmatic creation, discovery, and joining through standard RESTful interfaces. Teams can implement custom matchmaking logic while utilising Hathora's server orchestration, which helps prevent vendor lock-in at the game logic level.

Authentication integration effectively supports various enterprise scenarios. The platform manages anonymous players for guest access, custom authentication tokens for existing identity systems, and third-party identity provider integration. Session management operates seamlessly while providing secure room access controls essential for competitive multiplayer environments.

Connection routing capabilities automate optimal server endpoint discovery based on player location and room availability. This functionality reduces client-side networking complexity while ensuring optimal latency distribution, which is crucial for titles with global player bases requiring sub-100ms response times.

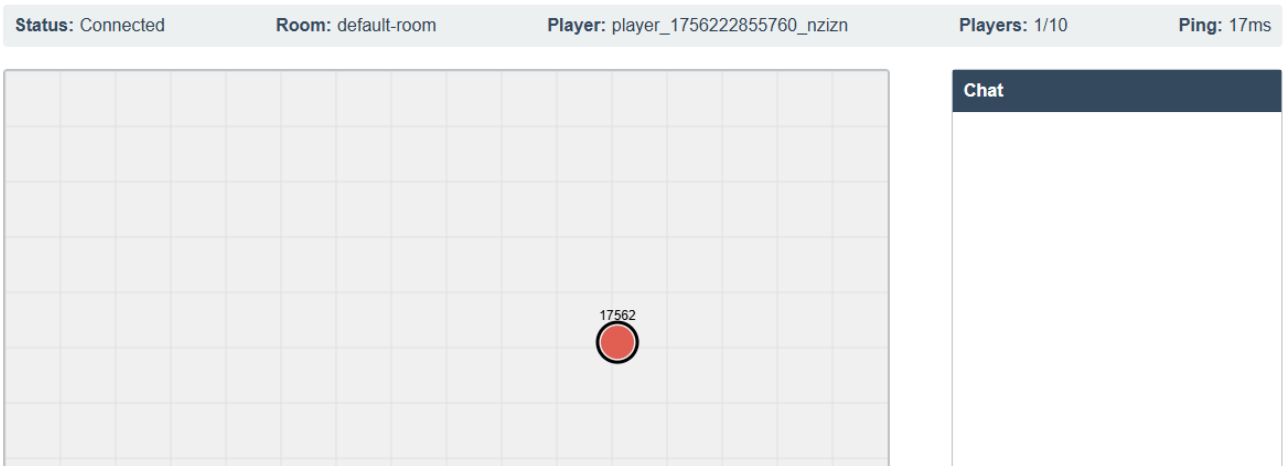
The matchmaker integration is particularly noteworthy for enterprise evaluation. The platform supports both simple room-based allocation and advanced matchmaking workflows. The CreateRoom API enables dynamic server provisioning with regional targeting, optimising latency through intelligent placement decisions that significantly impact player experience quality.

Regional optimisation leverages Hathora's Ping Service across fourteen global regions. Matchmaking systems can measure latency using UDP, ICMP, and WebSocket protocols before room creation, enabling data-driven server placement decisions. For global titles, this capability directly influences player retention metrics and competitive gameplay quality.

External matchmaker integration effectively meets enterprise architectural requirements. Teams can link Hathora rooms with external matchmaking systems through custom identifiers, integrating with existing backend infrastructure while utilising Hathora's server orchestration capabilities. This flexibility is crucial for organisations with established multiplayer technology stacks.

Testing confirmed the lobby service architecture's effectiveness through practical implementation. The three-tier pattern, where clients connect to lobby services that

provision Hathora rooms and direct player connections, maintains a clear separation between matchmaking logic and infrastructure management. For AAA projects, where performance consistency directly impacts player experience, dedicated hardware with account-level isolation ensures reliability for production deployments..



Multiplayer gameplay running on Hathora infrastructure showing sub-50ms regional latency

The platform SDK is available across major development environments: Unity, Unreal Engine, JavaScript/TypeScript, and native mobile platforms. Server-side integration requirements are minimal, typically involving environment variable configuration and optional API integration for enhanced functionality. This approach reduces integration friction while maintaining architectural flexibility.

The evaluation revealed comprehensive SDK documentation with practical integration examples. JavaScript SDK integration required fewer than fifty lines of code for basic room management functionality. Unity integration provided prefabs and example scenes that significantly accelerated development. The RoomV2Api integration demonstrated effective room lifecycle management with minimal implementation complexity, which is critical for teams with tight development timelines.

10 CI/CD integration and development workflow

Hathora's CI/CD capabilities address a key challenge for development teams: integrating multiplayer server deployments into automated workflows without major architectural changes. The platform's approach to continuous integration is especially valuable for teams familiar with modern development practices.

The CLI-based automation system integrates seamlessly with major CI platforms like TeamCity, Jenkins, and GitHub Actions through comprehensive tooling that supports scripted deployment workflows. This compatibility is crucial, as it prevents teams from needing to abandon existing CI investments for platform-specific solutions.

Complete build and deployment automation is achievable through simple CLI commands. Teams can script workflows from build creation to deployment activation, integrating with existing development pipelines without extensive modifications. This automation is vital for development cycles requiring frequent iteration and testing, particularly in multiplayer games where rapid iteration directly impacts development speed.

The build system architecture supports enterprise development practices through intelligent artifact management. Build reuse across multiple applications and environments within Hathora eliminates duplicate uploads and enables sophisticated environment management. The CLI integration requires minimal changes to existing deployment scripts and reduces adoption friction, which is a critical consideration for teams evaluating infrastructure migration costs.

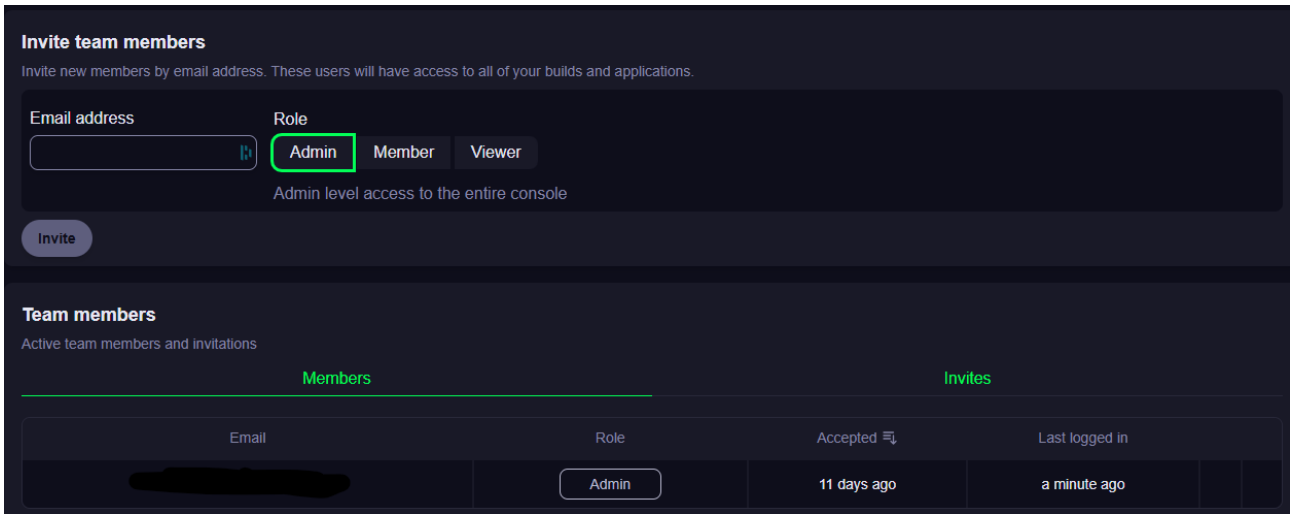
Container-first development workflows receive native support through Hathora's Docker-based deployment model. The platform automatically handles container orchestration while allowing teams control over build and deployment automation. This balance is essential for enterprise operations requiring both simplicity and developmental control.

Testing showed that integrating deployment pipelines requires significantly less effort than expected. Existing Docker-based workflows translate directly to Hathora deployment patterns, with minimal script modifications needed for automated deployment. This compatibility reduces migration risk and the learning curve for development teams.

11 Security considerations for development teams

Security requirements can burden game development teams, especially regarding enterprise compliance and protection against attacks that disrupt live games. Hathora's approach alleviates these concerns without requiring teams to become security experts.

The platform simplifies authentication using familiar methods like Google Sign-In alongside traditional credentials. Its role-based access system benefits game teams: admins manage billing and team oversight, developers gain deployment access without financial controls, and QA teams have read-only visibility for testing. This separation prevents junior developers from accidentally incurring costs or accessing production systems inappropriately.



For API integration, scoped tokens mitigate the risks of sharing full-access credentials across CI/CD systems and external services. Teams can generate tokens limited to specific operations such as builds, deployments, or monitoring, and they can revoke those tokens instantly if compromised. This granular approach enhances security without complicating development workflows.

DDoS protection is documented as a built-in feature, with baseline protections against common attack patterns for all customers. Enterprise customers have access to additional controls, such as providing trusted player IPs for rate-limiting during active attacks. We did not attempt to validate these protections in practice.

The platform resolves "noisy neighbour" issues in shared cloud infrastructure through complete compute isolation on higher tiers. For AAA projects, where performance consistency directly impacts player experience, dedicated hardware with account-level

isolation ensures reliability for production deployments. (– ADD if it is the shared tier you do share access to baremetal)

Builds upload over HTTPS to a signed URL and are stored in Hathora's private container registry. This approach reduces supply chain risks without requiring teams to implement custom security measures for their Docker workflows. Application logs are encrypted at rest and served over TLS, meeting enterprise security requirements while maintaining developer-friendly deployment processes.

12 Costs

Cost management is essential for multiplayer games, where player count fluctuations can lead to unexpected infrastructure costs. Hathora's hybrid approach tackles this challenge by combining predictable base costs with dynamic scaling.

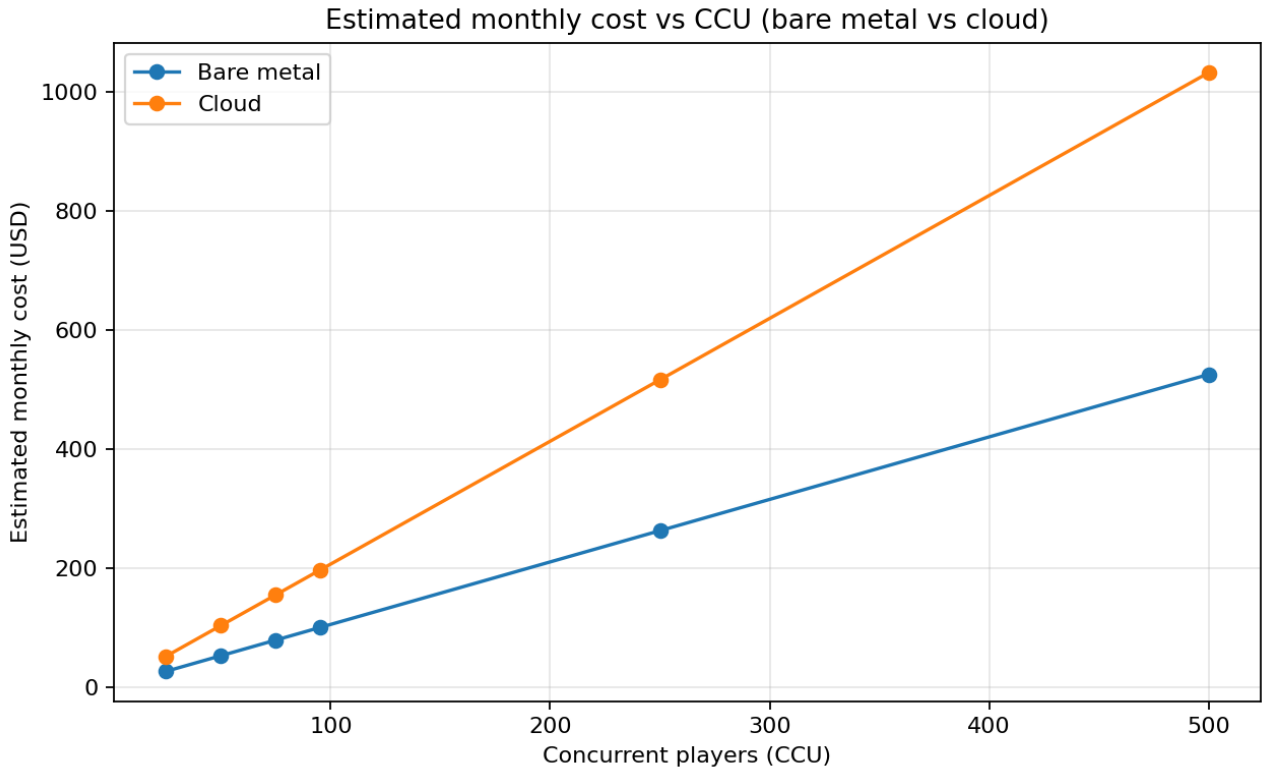
The platform integrates committed bare metal capacity with cloud overflow scaling, balancing cost efficiency and responsiveness to player demand. In principle, this can be cheaper than cloud-only hosting; in practice, savings depend on how efficiently the scaler and your matchmaker ramp capacity up and down, how quickly idle capacity is reclaimed, and your game's traffic profile. Cloud rates are highly competitive, so hybrid only beats cloud consistently when baseline utilisation is kept high on bare metal and burst windows are short (this specifically references cases where a game might use capacity lower than a single bare metal machine for any point during a 24 hour period in any one region), or if you manage to setup effective cloud scaling in Hathora that does not impact player experience. In our tests and sample workload, the hybrid model was cheaper, but results will vary by game and operations. Pricing is based on usage—charging for active vCPUs and outbound bandwidth. Bare metal capacity is reserved on a monthly commitment under the service terms, while overflow cloud usage remains metered and scales with demand.

Cost predictability improves with the one-month commitment model for bare metal capacity, allowing teams to adjust baseline capacity based on actual usage rather than long-term contracts. This is a month-to-month reservation, with changes typically taking effect at the next term; enterprise deployments often formalise baseline capacity and unit pricing in a contract. This flexibility is particularly valuable during game lifecycle changes, seasonal variations, or unexpected popularity shifts that can dramatically affect infrastructure needs.

For smaller deployments, under 100 concurrent players, cost advantages are clear. Even at larger scales, savings remain substantial, though teams should carefully model their expected ratio of base capacity to peak demand. Starting conservatively with bare metal allocation and expanding based on observed traffic patterns is the most effective strategy.

From a developer's perspective, the cost savings of Hathora hybrid model were evident in practice. In our runs we compared Hathora bare metal baseline to the same workload bursting on Hathora's cloud capacity and saw meaningful reductions on the bare-metal portion. This comparison is within Hathora's hybrid model (bare metal baseline versus cloud overflow), not against other providers. This aligns with Hathora's guidance that, for the same server class, bare metal can be more than 60% cheaper than equivalent cloud capacity. The optimal split between bare metal and cloud is game-specific: pricing is based on active vCPUs and outbound bandwidth, so how well the system keeps sustained load on bare metal depends on your game's CPU-per-player,

bandwidth-per-player, average session length, and traffic shape. Titles with steadier, CPU-bounded rooms tend to realise a higher bare-metal share, while spiky or bandwidth-heavy workloads will burst to cloud more often. Importantly, these cost savings do not come at the expense of security. DDoS protections, null routing, and network controls remain in place for both bare metal and cloud capacity.



Monthly cost vs CCU — Hathora bare metal vs cloud overflow

Scaling performed well under real traffic conditions. By default, it targets 75-85% utilisation, prioritising cheaper bare metal and only bursting to cloud resources when necessary. When demand decreased, it quickly removed underutilised cloud capacity, with new cloud capacity coming online in just under two minutes.

You can maintain predictable costs by setting small cloud minimums in key regions for faster cold starts and relying on bare metal for steady loads. Hathora's documentation notes that allocations happen very quickly when capacity exists (often within ~5s), so the platform is generally able to handle spikes of tens to a few hundred new players without intervention; very large, sudden surges (thousands of new players in seconds) are best handled with pre-warmed capacity, regional cloud minima, or explicit pre-scaling for events.

13 Evidence

Our hands-on testing of the Lyra server deployment revealed practical insights into the platform's performance. Regional connections consistently delivered response times under 50ms, meeting the latency requirements for most multiplayer game genres. The deployment pipeline, from build upload to active server status, completed in under 2 minutes, which is sufficient for iterative development but slower than ideal for emergency hotfixes.

The automatic scaling behaviour responded effectively during load testing, typically adjusting capacity within 2-3 minutes of demand shifts. This response time suits most game traffic patterns, although games with highly volatile player counts may face brief capacity constraints during rapid scaling.

Architecturally, the platform's networking implementation supports highly connected multiplayer scenarios. Server instances can connect to external services and communicate with other game servers without restrictions, enabling sophisticated distributed architectures. The containerised environment imposes no networking limitations, essential for cross-server player transfers and distributed game state management.

By default, outgoing traffic from containers is not restricted by Hathora or Docker, which is convenient for integrations but may be a drawback for teams needing strict outbound controls; additional filtering must be implemented inside the container or at the network level.

The developer experience exceeded expectations in several areas. Documentation quality is consistently high, and support channels respond promptly to technical inquiries. Both the CLI and web console interfaces offer adequate functionality for build management and fleet operations. Teams familiar with containerisation can typically complete the setup process, from account creation to first deployment, within an hour.

Automated fleet management significantly reduces operational overhead compared to managing server infrastructure manually on traditional cloud providers. This simplification translates to meaningful time savings for development teams, allowing them to focus on game logic rather than infrastructure maintenance.

Performance monitoring capabilities provide visibility into server health and player connection metrics. The logging system captures relevant game server events, though the log format and filtering options could improve to better support debugging complex multiplayer issues. Despite these limitations, the platform delivers on its core promise of simplified multiplayer infrastructure management while maintaining the networking flexibility required for advanced game architectures.

14 Conclusion

Hathora delivers on its core promise: it makes standing up and running multiplayer servers fast, approachable, and low friction. The hybrid cloud model, combining bare metal baseline capacity with cloud overflow, provides cost efficiency while maintaining responsiveness to demand spikes. The rooms model maps cleanly to how games are built, the docs and tooling make CI/CD straightforward, and observability is practical out of the box. In testing, we moved from zero to live servers quickly and iterated without wrestling infrastructure.

There are real trade offs. At the time of writing, while there are no proven AAA, high CCU or high network egress launches on the platform, Hathora has successfully demonstrated real-world reliability and operational maturity through its existing launches, case studies of which are listed here.

The abstraction that simplifies operations also limits deep system tuning and custom networking, storage, or security controls. Fleet management with dedicated hardware is available across all paid plans, while bring your own cloud (BYOC) requires the enterprise tier, and regional coverage is good but not as broad as it could be. Teams with large scale, strict compliance, or heavy in house Ops support may find these constraints material.

For smaller teams looking to reduce time to market or larger teams seeking a strong out of the box solution, Hathora is a good fit. It is easy to set up, simple to use day to day, and supports rapid iteration while maintaining reliability. Teams with extensive customisation requirements or specialised infrastructure needs should validate their requirements through thorough testing. If you value developer speed and clear workflows over maximum customisation and tuning, Hathora is an excellent option.

15 References

Hathora Cloud Documentation. Hathora, 2025. https://hathora.dev/docs/

How Hathora Works - Architecture and Core Concepts. Hathora, 2025. https://hathora.dev/docs/category/how-hathora-works

Builds and Deployments Guide. Hathora, 2025. https://hathora.dev/docs/how-hathora-works/builds-deployments

Fleets and Autoscaling. Hathora, 2025. https://hathora.dev/docs/how-hathora-works/fleets-autoscaling

Telemetry and Monitoring. Hathora, 2025. https://hathora.dev/docs/how-hathora-works/telemetry

Security Features. Hathora, 2025. https://hathora.dev/docs/how-hathora-works/security

Scoped Permissions Guide. Hathora, 2025. https://hathora.dev/docs/guides/scopes

CI/CD Integration Guide. Hathora, 2025. https://hathora.dev/docs/guides/ci-cd

Deployment Quick Start. Hathora, 2025. https://hathora.dev/docs/guides/deploy-hathora

Docker Configuration Guide. Hathora, 2025. https://hathora.dev/docs/guides/create-dockerfile

Developer Token Generation. Hathora, 2025. https://hathora.dev/docs/guides/generate-developer-token

API Reference Documentation. Hathora, 2025. https://hathora.dev/api

Hathora GitHub Organisation. GitHub, 2025. https://github.com/hathora

TypeScript SDK. GitHub, 2025. https://github.com/hathora/cloud-sdk-typescript

Unity SDK. GitHub, 2025. https://github.com/hathora/cloud-sdk-unity

Java SDK. GitHub, 2025. https://github.com/hathora/cloud-sdk-java

Go SDK. GitHub, 2025. https://github.com/hathora/cloud-sdk-go

Godot Plugin. GitHub, 2025. https://github.com/hathora/hathora-godot-plugin

Hathora Console. Hathora, 2025. https://console.hathora.dev/

Pricing Information. Hathora, 2025. https://hathora.dev/pricing

Manifesto

Don't Be a Dick - we respect and love our partners, industry colleagues, and teammates. ***Everyone is Awesome*** - our worldview is inclusive. We are open and welcoming. No Code Wizard is more important than another. There are no barriers. ***Freedom of speech*** is a fundamental right for all.

We ***foster partnerships***, not supplier/client relationships- our foundation is mutual trust and value creation. We operate as a ***unified team***, not individual guns for hire - every challenge is shared and solved collectively. ***Easy to work with***; hard to surprise - we apply the ideal people and the right tech to each project. Our specialist wizards are ***battle-hardened veterans*** - experienced and unphased by even the most unexpected challenges.

Everything we do flows from this.